News

Quiz next Wednesday

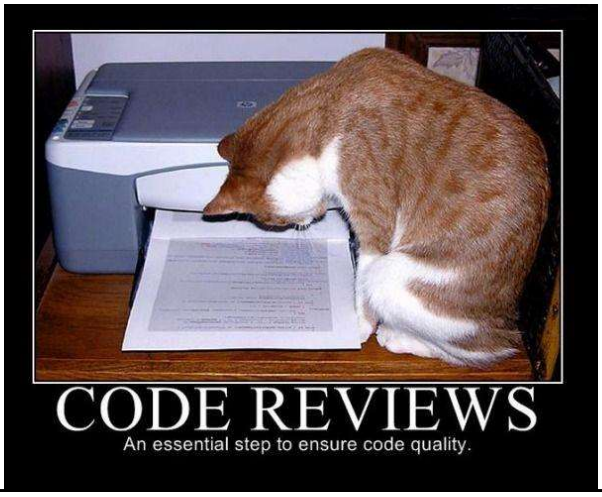
I have a LOT of previous quizzes. Please drop by before leaving today, during lab time

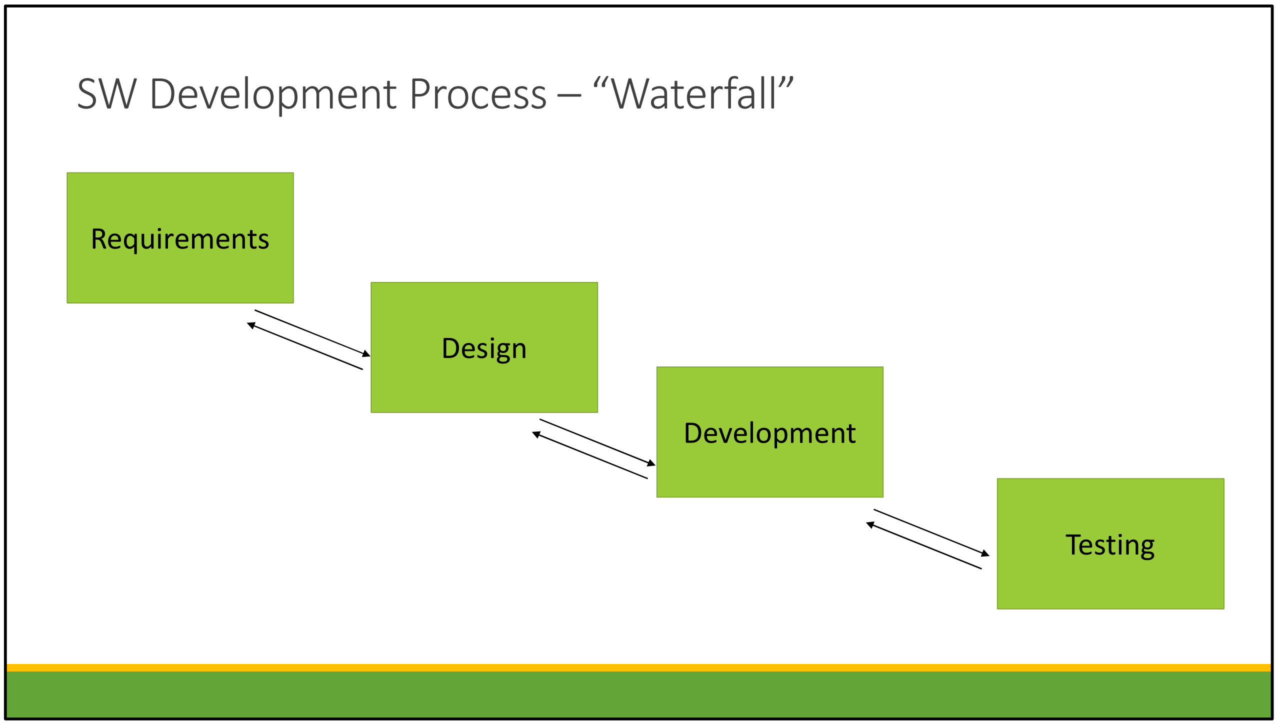
Happy Thanksgiving!

CS112 –
Java
Programming

Fall 2022

Professional Software Development Process

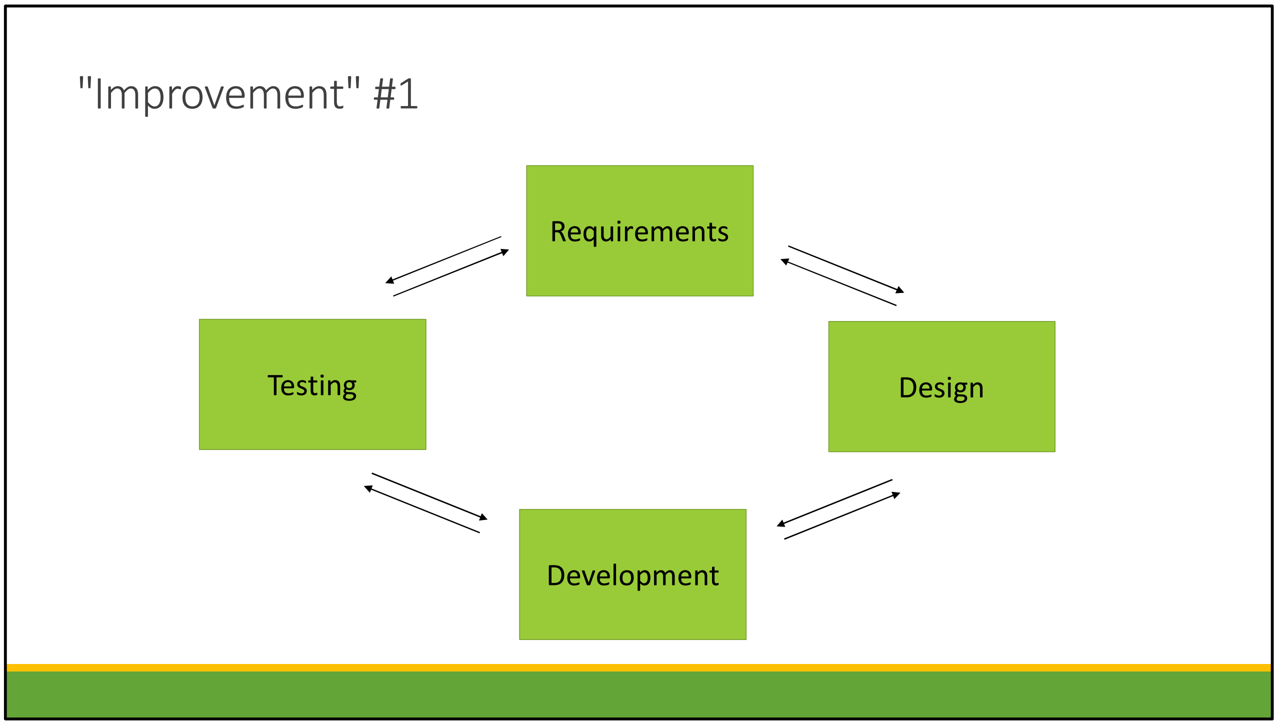




SW Development Process – “Waterfall”

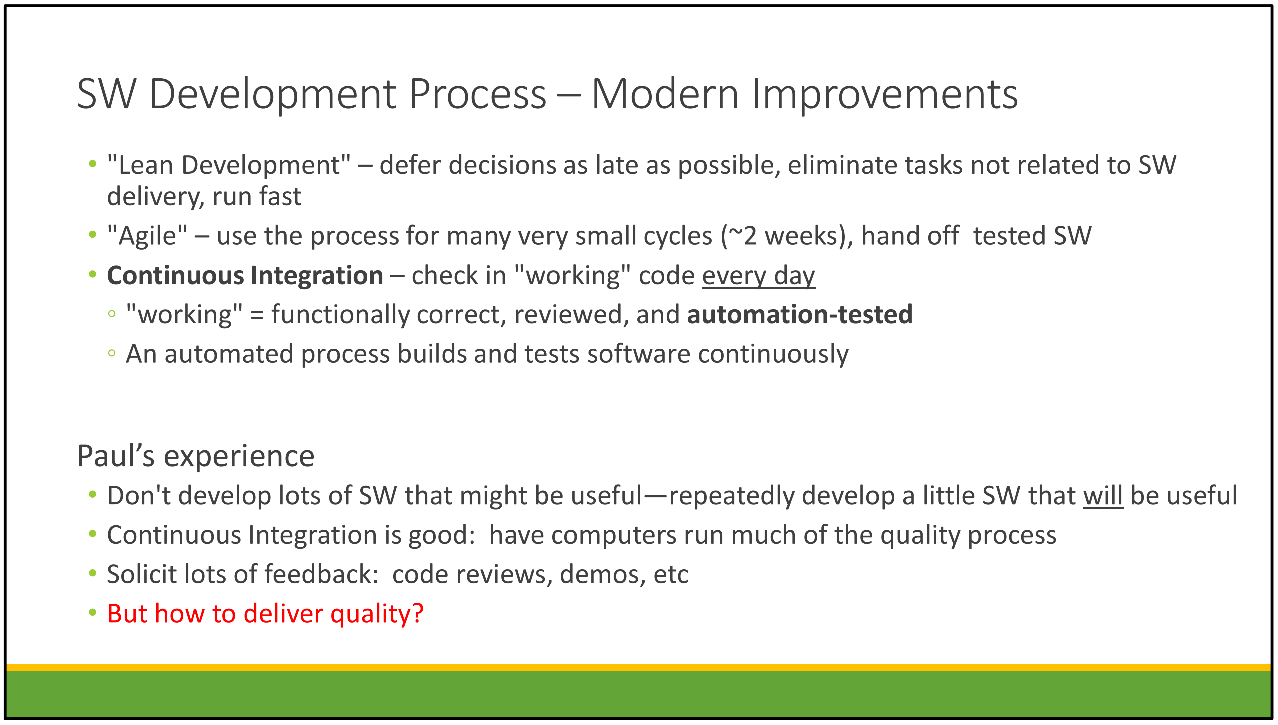
Requirements
Design
Development
Testing

"Waterfall" model: fall from one stage to next
Reality: requirements change, priorities change, Development experience improves design, Testing changes everything



"Improvement" #1

Modified Waterfall: Reflects reality but can be slow and worse unpredictable. Costly



SW Development Process – Modern Improvements

- "Lean Development" – defer decisions as late as possible, eliminate tasks not related to SW delivery, run fast
- "Agile" – use the process for many very small cycles (~2 weeks), hand off tested SW
- **Continuous Integration** – check in "working" code <u>every day</u>
  - "working" = functionally correct, reviewed, and **automation-tested**
  - An automated process builds and tests software continuously

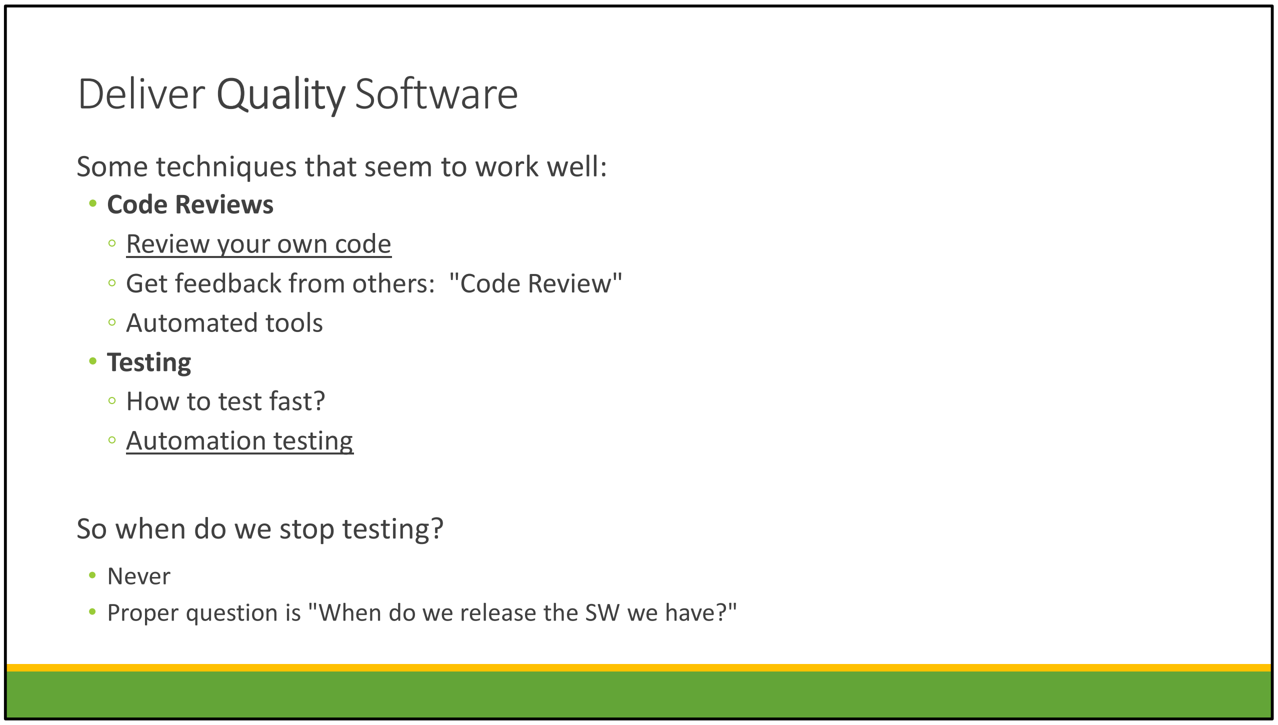
Paul's experience

- Don't develop lots of SW that might be useful—repeatedly develop a little SW that <u>will</u> be useful
- Continuous Integration is good: have computers run much of the quality process
- Solicit lots of feedback: code reviews, demos, etc
- But how to deliver quality?

Activities not related to SW development: training, documentation, big project reviews

1) Agile great but depends on having end customers who are smart about requirements and able to give feedback every 2 weeks

Not often practical. But still some very useful improvements.
BIG DEAL #1: ONE TEAM responsible for understanding requirements, developing SW, testing SW, supporting SW. No divided responsibilities.
BIG DEAL #2: DEVELOP SMALL PIECES OF SW. TEST THEM. DEMO THEM. GET FEEDBACK. MOVE ON.
RISK: In my experience "Agile" too often means "let the end customer test and then fix bugs fast". ok for web, lousy for space shuttle.



Deliver Quality Software

Some techniques that seem to work well:

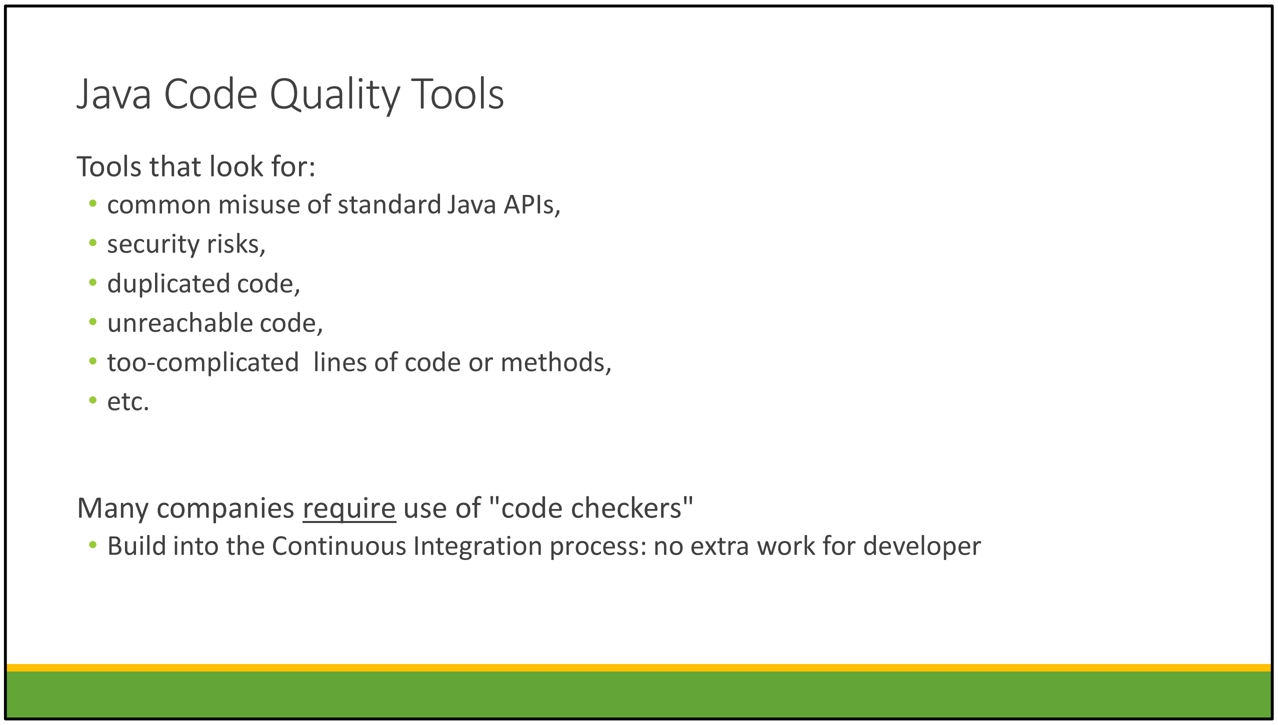
- **Code Reviews**
  - Review your own code
  - Get feedback from others: "Code Review"
  - Automated tools
- **Testing**
  - How to test fast?
  - Automation testing

So when do we stop testing?

- Never
- Proper question is "When do we release the SW we have?"

1) In professional enviro, code reviews are required for EVERY SINGLE SW CHECKIN
2) In professional enviro, automation testing runs continuously, 3x/day.
3) Tools monitor 'code coverage'. Teams track and manage where bugs occur (SW areas, teams)
4) Teams spend as much time developing test SW as product SW. Still much more productive than testing manually. Which is much more productive than having customers test.

Many big companies release SW every day
My smallish employer issued 1000 customer releases per year (many customer-specific).
Need fast AND high quality



Java Code Quality Tools

Tools that look for:

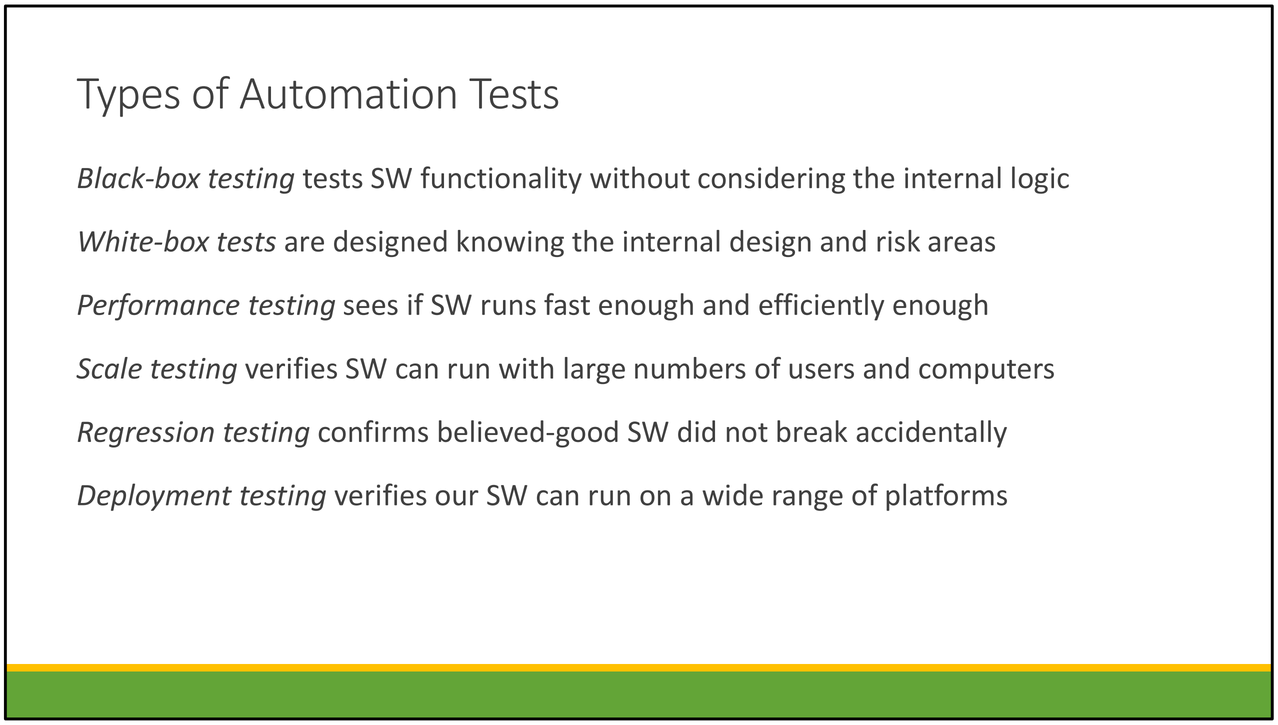
- common misuse of standard Java APIs,
- security risks,
- duplicated code,
- unreachable code,
- too-complicated lines of code or methods,
- etc.

Many companies require use of "code checkers"

- Build into the Continuous Integration process: no extra work for developer

There are MANY TOOLS
Let's do a TYPE-ALONG:

- Start Eclipse. Help->Eclipse Marketplace
- Search for Sonar. Install SonarLint
- Look at right edge of code window for little blue dashes. Hover: comments on the code!



Types of Automation Tests

Black-box testing tests SW functionality without considering the internal logic

White-box tests are designed knowing the internal design and risk areas

Performance testing sees if SW runs fast enough and efficiently enough

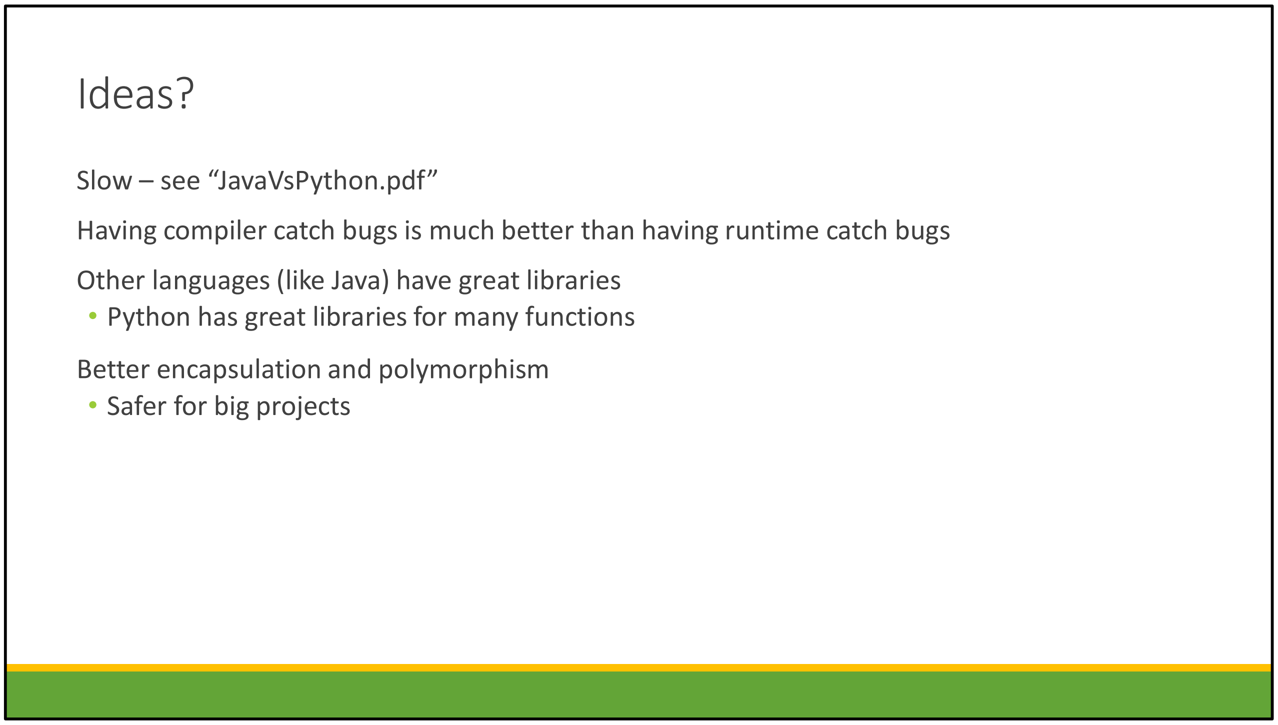
Scale testing verifies SW can run with large numbers of users and computers

Regression testing confirms believed-good SW did not break accidentally

Deployment testing verifies our SW can run on a wide range of platforms

First, Test Developer is highly skilled career: Software Developer Engineer in Test
All these tests are vital.
My experience: same SW ran on "bundled appliance", customer computer server, and public cloud.
Lots of professional tools to help testing: code risks like accessing null references, reading past end of arrays, memory growth

Interlude: why doesn't everyone just use Python for everything?



Ideas?

Slow – see “JavaVsPython.pdf”

Having compiler catch bugs is much better than having runtime catch bugs

Other languages (like Java) have great libraries

- Python has great libraries for many functions

Better encapsulation and polymorphism

- Safer for big projects

Paul's work experience

- Products had to run continuously for 8+ years
- Advisor’s SETI project: SW must run for 1000+ years